AI for the Web — Ontology-based Community Web Portals

Steffen Staab[a,b], Jürgen Angele[b], Stefan Decker[a,b], Michael Erdmann[a], Andreas Hotho[a], Alexander Maedche[a], Hans-Peter Schnurr[a,b], Rudi Studer[a,b], York Sure[a]

email: {staab, angele, decker, erdmann, hotho, maedche, schnurr, studer, sure}@aifb.uni-karlsruhe.de
[a]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS
[b]ontoprise GmbH, Hermann-Löns-Weg 19, 76275 Ettlingen, Germany
http://www.ontoprise.com

Abstract

Community web portals serve as portals for the information needs of particular communities on the web. We here discuss how a comprehensive, ontology-based approach for building and maintaining a high-value community web portal has been conceived and implemented. The ontology serves as a semantic backbone for accessing knowledge on the portal, for contributing information, as well as for developing and maintaining the portal. In particular, the ontology allows for flexible querying and inferencing of knowledge. Actual usage of our technology is facilitated through a set of tools that are about to turn our research system into a portal for wide-spread usage right now. The development of these tools has greatly benefited from some first experiences we had with actual users of the community web portal of the knowledge acquisition community.

1 Introduction

One of the major strengths of the World Wide Web is that virtually everyone who owns a computer may contribute high-value information — the real challenge is to make valuable information be found. Search machines help with this task, but ultimately they fail to provide appropriatly structured views onto the web.

From the very beginning of the web, communities of interest have formed that covered what they deemed to be of interest to their members in — what we here call — community web portals. Community web portals are similar to Yahoo!™ and its likes by their goal of presenting a structured view onto the web, however they are dissimilar by the way knowledge is provided in a collaborative process with only few resources (manpower, money) for maintaining and editing the portal. Thus, their is a need for automation of management of community web portals.

Community web portals try to weave loose pieces of information into a coherent presentation adequate for sharing knowledge with the user. Support through ontologies appears as an appropriate means in order to facilitate the tool-supported structuring of knowledge. The ontology formally represents common knowledge and interests that people share within their community. It is used to support the major tasks of a portal, viz. accessing the portal through manifold, dynamic, conceptually plausible views onto the information of interest in a particular community (Section 2), and providing information in a number of ways that reflect different types of information resources held by the individuals (Section 3). The subsequent Section 4 shows how an ontology-based web portal may be developed and maintained using our set of methods and tools and how the overall architecture of the portal looks like. Before we conclude, we compare our work with related approaches (Section 5).

The Example. The example that we draw from in the rest of this paper is the portal for the *"Knowledge Annotation Initiative of the Knowledge Acquisition community"* (KA2; *cf.* (Benjamins, Fensel, & Decker 1999)). The KA2 initiative has been conceived for semantic knowledge retrieval from the web building on knowledge created in the KA community. To structure knowledge, an ontology has been built in an international collaboration of researchers. The ontology constitutes the basis to annotate WWW documents of the knowledge acquisition community in order to enable intelligent access to these documents and to infer implicit knowledge from explicitly stated facts and rules from the ontology.

Given this basic scenario, which may be easily transferred towards other settings for community web portals, we have investigated the techniques and built the tools that we describe in the rest of this paper. Nevertheless the reader may note that we have not yet achieved a complete integration of all tools and neither have we exploited all our technical capabilities in our up and running demonstration KA2 community web portal (http://ka2portal.aifb.uni-karlsruhe.de).

2 Accessing the Community Web Portal

Navigating through a, maybe unknown, portal is a rather difficult task in general. Information retrieval may of course help, but it may also be more of a hindrance, because the user may not know the conceptualization that underlies the portal. Hence, we provide query and navigating capabilities and make the conceptual background transparent to the user.

Our description of access capabilities in this section starts with the query capabilities of our representation framework. The framework builds on the very same F-Logic (Kifer, Lausen, & Wu 1995) mechanism for querying as it does for ontology representation and, thus, it may also exploit and explicate the ontological background knowledge. In addition to these facilities for explanation and exploration, the ontology also acts as a mediator between proprietary informa-

tion sources that provide additional background knowledge to the portal (cf. (Wiederhold & Genesereth 1997) on mediators). Nevertheless, F-Logic is as poorly suited for presentation to naive users as any other query language. Hence, its use is mostly disguised in various easy-to-use mechanisms that more properly serve the needs of the common user (*cf.* Section 2.2), while it still gives the editor all the power of the principal F-Logic representation and query capabilities. Finally in this section, we touch upon some very mission-critical issues of the actual inference engine that answers queries and derives new facts by combining facts with structures and rules from the ontology.

2.1 Query Capabilities

Though information may be provided in a number of different formats our underlying language for representation and querying is F-Logic. For instance, using a concrete example from our showcase the following query asks for all publications of the researcher "Steffen Staab".

(1) `FORALL` $Pub \leftarrow$
`EXISTS` ResID ResID:*Researcher* [NAME $\twoheadrightarrow$ "Steffen Staab"; PUBLICATION $\twoheadrightarrow$ Pub].

The substitutions for the variable *Pub* are the desired publications. The expressiveness and usability of such queries is strongly increased by the possibility to use a simple form of information retrieval using regular expressions whithin queries.

In addition, the query capabilities enable using the background knowledge expressed in the KA2 ontology using rules. For example, one rule states that, two researchers cooperate, if a *Researcher* X works at a *Project* $Proj$ and if a *Researcher* Y works at the same *Project* $Proj$ and X is another person than Y. The rule is formulated in F-Logic as follows:

(2) `FORALL` $X, Y, Proj$
X : *Researcher* [COOPERATESWITH $\twoheadrightarrow$ Y : *Researcher*]
$\leftarrow$
X : *Researcher* [WORKSATPROJECT $\twoheadrightarrow$ $Proj$: *Project*]
`AND`
Y : *Researcher* [WORKSATPROJECT $\twoheadrightarrow$ $Proj$: *Project*]
`AND NOT equal`(X, Y).

If we take a look at web pages about research projects, typically information about the researchers (*e.g.* their names, their affiliation, ...) involved in the projects is explicitly stated in HTML. However, the fact that researchers who are working together in projects are cooperating is not explicitly stated on the web pages. A question might be: "Which researchers are cooperating with other researchers?" Querying for cooperating researchers the implicit information about project cooperation of researchers is retrieved. The query may be formulated using F-Logic:

(3) `FORALL` ResID1, ResID2 $\leftarrow$
ResID1: *Researcher* [COOPERATESWITH $\twoheadrightarrow$ ResID2].

The result set includes explicit information about a researchers cooperation relationships, which are stored in the knowledge warehouse, and also implicit information about project cooperation between researchers derived using the project-cooperation rule modeled in (2)..

2.2 Navigating and Querying the Portal

Usually it is too inconvenient for users to query the portal using F-Logic. Therefore, we offer a range of techniques that allow for navigating and querying the community web:

- A *hypertext link* may contain a query which is dynamically evaluated when one clicks on the link. Browsing is realized by defining views on the top-level concepts of the KA2 ontology, such as *Persons*, *Projects*, *Organizations*, *Publications*, *Research Topics* and *Events*. For example clicking on the *Projects* hyperlink results in a query for all projects known to the portal. The query is evaluated and the results are presented to the user in a table.
- A choice of concepts, instances, or combinations of both may be issued to the user in *HTML forms*. Choice options may be selected through check boxes or radio buttons. For instance, clicking on the *Projects* link (*cf.* upper part of Figure 1) an F-Logic query is evaluated and all projects contained in the portal are retrieved. The results can be restricted using topic-specific attributes contained in the KA2 ontology for projects, such as topics of a project, people involved etc. The selection list (*e.g.* for all people involved in projects) is generated dynamically from the information contained in the knowledge warehouse (*cf.* Section 3.4).

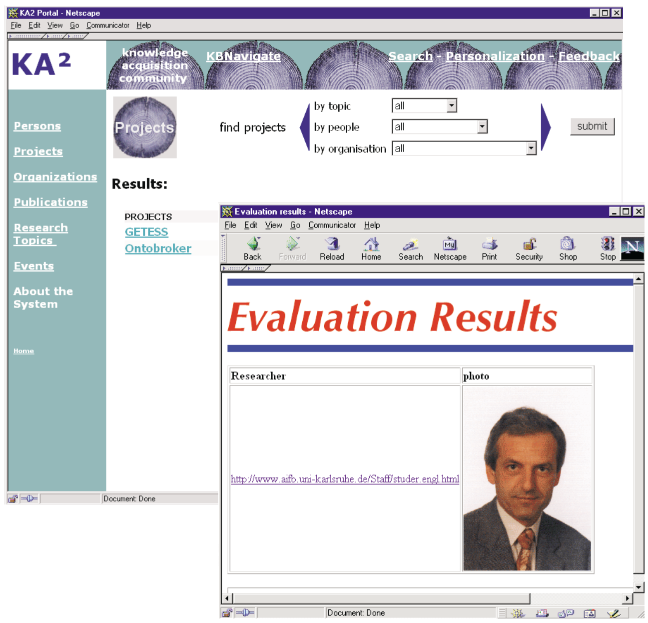

Figure 1: Accessing the Community Web Portal

- A query may also be generated by using the *hyperbolic view interface*. The hyperbolic view visualizes the ontology as a hierarchy of concepts. The presentation is based on hyperbolic geometry (*cf.* (Lamping, Rao, & Pirolli 1995)). When a user selects a node from the hyperbolic concept hierarchy, a form is presented which allows the user to select attributes or to insert values for the attributes. A result of the user request searching for the community member "Rudi Studer" and his photo is shown

in the left part of Figure 1. Based on the selected node and the corresponding attributes, a query is compiled.

- Furthermore, queries created by the hyperbolic view interface may be stored using the personalization feature. Queries are personalized for the different users and are available for the user in a selection list. The stored queries can be considered as *semantic bookmarks*. By selecting a previously created bookmark, the underlying query is evaluated and the updated results are presented to the user. By this way, every user may create a personalized view on the portal.
- Finally, we offer an expert mode. The most technical (but also most powerful and flexible) way for querying the portal requires that F-Logic is typed in by the user. This way is only appropriate for users who are very familiar with F-Logic and the KA2 ontology.

2.3 The Inference Engine

The inference engine answers queries and it performs derivations of new knowledge by an combination of facts and the ontology. While the expressiveness of F-Logic and its Java powered realization in our inference engine is one of the major arguments for using it in a semantic community web portal, wide acceptance of a service like this also depends on *prima facie* unexciting features like speed of service. The principal problem we encounter here is that there exist worst case situations (not always recognizable as such by the user) where a very large set of facts must be derived by the inference engine in order to solve a particular query. While we cannot guarantee for extremely fast response times in all cases, unless we drastically cut back on the expressiveness of our representation formalism, we provide several strategies to cope with performance problems:

- The inference engine may be configured to subsequently deliver answers to the query instead of waiting for the entire set of answers before these answers are presented to the user. This has the consequence that answers which are directly available as facts may be presented immediately while other anwers which have to be derived using rules are presented later.
- The inference engine caches all facts and intermediate facts derived from earlier queries. Thus, similar queries or queries that build on previously derived facts may be answered fast.
- Finally, we allow the inference engine to be split into several inference engines that execute in parallel. Every engine may run on a different processor or even a different computer. Every inference engine administers a subset of the rules and facts. A master engine coordinates user queries and distributes subqueries to the slave engines. These slave engines either answer these subqueries directly or distribute incoming subqueries to other inference engines.

3 Providing Information

An essential feature of a community web portal is the contribution of information from all (or at least many) members of the community. Though they share some common understanding, the pieces of information they may contribute may come in many different (legacy) formats. Hence, one needs a set of methods and tools that may account for the diversity of information sources of potential interest to the community portal. These methods and tools must be able to cope with different syntactic mechanisms and they must be able to integrate different semantic formats based on the common ontology.

Considering the syntactic and/or interface side, we support three major, different, modes of information provisioning: First, we handle *metadata-based information sources* that explicitly describe contents of documents on a semantic basis. Second, we align regularities found in documents or data structures with the corresponding semantic background knowledge in *wrapper-based* approaches. Third, we allow the direct provisioning of facts through our *fact editor*. All the information is brought together in the knowledge warehouse. Thus, it mediates between the original heterogeneous information sources.

3.1 Metadata-based Information

Metadata-based information enriches documents with semantic information by explicitly adding metadata to the information sources. Over the last years several metadata languages have been proposed which can be used to annotate information sources. Current web standards can be handled within our semantic web portal approach. On the one hand, RDF (W3C 1999) facts serve as direct input for the knowledge warehouse, on the other hand, RDF facts can be generated from information contained in the portal knowledge warehouse. We developed *SiLRI* (Simple Logic-based RDF Interpreter), a logic-based inference engine implemented in Java that can draw inferences based on the RDF data model (Decker *et al.* 1998). XML provides the chance to get metadata for free, *i.e.* as a side product of defining the document structure. For this reason, we have developed a method and a tool called *DTDMaker* for generating DTDs out of ontologies (Erdmann & Studer 1999). DTDMaker derives an XML document type definition from a given ontology in F-Logic, so that XML instances can be linked to an ontology. The linkage has the advantage that the document structure is grounded on a true semantic basis and, thus, facts from XML documents may be directly integrated into the knowledge warehouse.

HTML-A, our HTML extension, adds annotations to HTML documents using an ontology as a metadata schema. HTML-A has the advantage to smoothly integrate semantic annotations into HTML and prevents the duplication of information. To facilitate the annotation of HTML, we have developed an HTML-A annotation tool called *OntoPad*.

3.2 Wrapper-based Information

In general, annotating information sources by hand is a time consuming task. Often, however, annotation may be automated when one findes regularities in a larger number of documents. The principle idea behind wrapper-based information is that there are large information collections that have a similar structure. We here distinguish between semi-structured information sources (*e.g.* HTML) and structured information sources (*e.g.* relational databases).

Semi-structured Sources. In recent years several approaches have been proposed for wrapping semi-structured documents, such as HTML documents. Wrapper factories (*cf.* (Sahuguet & Azavant 1999)) have considerably facilitated the task of wrapper construction. In order to wrap directly into our knowledge warehouse we have developed our own wrapper approach OntoWrapper that directly aligns regularities in semi-structured documents with their corresponding ontological meaning.

Structured Sources. Though, in the KA2 community web there are no existing information systems, we would like to emphasize that existing databases and other legacy-systems may contain valuable information for building a community web portal.

3.3 Fact Editor

The process of providing new facts into the knowledge warehouse should be as easy as possible. For this reason we offer the *Fact Editor*, which is another mode in which the hyperbolic interface tool may be used. At this time the forms are not used to ask for values, but to introduce values for attributes of instances of a corresponding concept from the ontology. The Fact Editor is also used for maintaining the portal to add, modify, or delete facts.

3.4 Knowledge Warehouse

The different methods and tools we have just described feed directly into the knowledge warehouse or indirectly when they are triggered by a web crawl. The warehouse itself hosts the ontology, *i.e.* the metadata level, as well as the data proper. The knowledge warehouse is indirectly accessed, through a user query or a query by an inference engine such as described in Section 2. Hence, one may take full advantage of the distribution capabilities of the inference engine and, likewise, separate the knowledge warehouse into several knowledge bases or knowledge marts. Facts and concepts are stored in a relational database, however, they are stored in a *reified* format that treats relations and concepts as first-order objects and that is therefore very flexible with regard to changes and amendments of the ontology.

4 Development of Web Portals

4.1 The Development and Maintenance Process

Even with the methodological and tool support we have described so far, developing a web portal for a community of non-trivial size remains a complex task. Strictly ad-hoc rapid prototyping approaches easily doom the construction to fail or they easily lead up to unsatisfactory results. Hence, we have thought about a more principled approach towards the development process that serves as means for documenting development, as well as for communicating principal structures to co-developers and editors of the web portal. We distinguish different phases in the development process that are illustrated in Figure 2. For the main part this model is a sequential one. Nevertheless, at each stage there is an evaluation as to whether and as to how easily further development may proceed with the design decisions that have been accomplished before. The results feed back into the results of earlier stages of development.

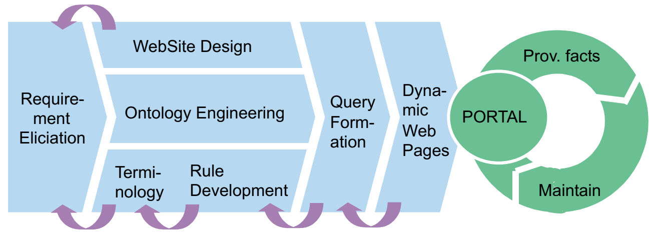

Figure 2: The Development Process of the Community Web Portal

The main stages of the development process and their key characteristics are given in the following:

- The process starts with the *elicitation* of user requirements in the requirements elicitation phase. In this phase, requirements about important and interesting topics in the domain are collected, the information goals of potential users of the portal are elicited, and preferences or expectations concerning the structure and layout of presented information is documented. Results of this very first phase constitute the input for the design of the web site and for preliminary HTML pages and affect the formal domain model embodied in the ontology.
- The requirements determine, *e.g.*, which views and queries are useful for users of the portal, which navigation paths they expect, how different web pages are linked, or which functionality is provided in different areas of the portal. Requirements like these are realized in the *web site design*. This design phase may be performed independently to a very large extent from the underlying formal structuring, *i.e.* the ontology.
- In parallel to the development of the structure and layout of the web site an *ontology engineering* process is started. The first phase elicits relevant *domain terms* that need to be refined and amended in the ontology engineering phase. First, the static ontology parts, *i.e.* the concept hierarchy, the attributes, and relations between concepts are formally defined. Thereafter, *rules* and constraints are developed. Rule development may incur a major revision of the concept hierarchy as consequence.
- In the *query development* step the views and queries described in one of the earlier phases are formalized. At first, their functionality is tested independently from the web site design. To express the information needs formally, the developer has to access the ontology, whereby additional rules or relations that define new views or ease the definition of queries may become necessary.
- Finally, web pages are populated, *i.e.* the queries and views developed during website design, and formalized and tested during query formalization are integrated into the operational portal. Information may be accessed via the portal as soon as a sufficient amount has been made available as outlined in Section 3.

During operation of the community portal it must be fed and maintained:

- The user community provides facts via numerous input channels (*cf.* Section 3).

- These facts may contain errors or undesired contents, or the integration of different sources may lead to inconsistencies. To counter problems like these a person is responsible to detect these cases and act appropriately. The detection of inconsistencies is supported by the inference engine via constraints formulated in F-Logic. The editor then has to decide how to proceed. He may contact responsible authors of conflicting information, he may simply ignore it, or he may manually edit the information.
- Changing requirements of the community must be reflected in the portal, *e.g.* popularity increasing in new fields of interests or technologies or viewpoints that shift may incur changes to the ontology, new queries, or even a new web site structure. In order to meet such new requirements, the above mentioned development process may have to be partially restarted.

4.2 Tools for Development and Maintenance

The previous subsection has described the principal steps for developing a community web portal. For efficient development of a community web portal, however, the process must be supported by tools. In the following, we describe the most important tools that allow us to facilitate and speed up the development and maintenance process. The tools cover the whole range from ontology engineering (OntoEdit), query formulation (Query Builder), up to the creation of dynamic web pages with the help of HTML/JavaScript templates. An overview of the tool suit is given in Table 1.

Table 1: Tool Suit for Community Portals

Tool	Description
OntoEdit	Ontology Engineering Environment
OntoPad	HTML Annotation Tool
Inference Engine	Reasoning service for query answering
OntoWrapper	Extracting information from semi-structured documents
Query Builder	Visual creation of queries
Fact Editor	Manual provision and maintenance of facts
Knowledge Warehouse	Fact base of the community web portal
HTML and JavaScript templates	Support the development of virtual HTML pages
Hyperbolic View	Visual querying the community web portal

OntoEdit. OntoEdit is a Ontology Engineering Environment delivering a wide range of functionalities for the engineering of ontologies including the modeling of concepts, relations, rules, and general ontology metadata (cf. (Maedche *et al.* 2000)). It includes several views on the modeling primitives that enable the user to state common legalities (*e.g.* the symmetry of the cooperation relationship between two persons).

Query Builder. While queries with low complexity can be expressed in F-Logic using the rule debugger alone, in other cases it is more convenient to create queries using our Query Builder tool. Such queries may then be integrated as links in a web page within a web editor by copying and pasting it into the web editors form. The Query Builder also contains a rule debugger, providing different views on different levels of detail to the proof tree that visualize relevant rules and rule parts for tracing the derivation process. In addition, generated queries by the Query Builder can be directly embedded into HTML/JavaScript Templates.

HTML/JavaScript Templates. Another time consuming activity is the development of the web pages that assemble queries from parts and that display the query results. For that purpose we have developed a library of template pages:

- Templates with check boxes, radio boxes, and selection lists are available. These HTML forms produce data which are used in Javascript functions to generate queries.
- The results of a query are fed into a template page as Javascript arrays. From these data different presentation forms may be generated:
  - A general purpose template contains a table that presents answer tuples returned by the inference engine in a HTML table. The template provides functions to sort the table in ascending or descending order on different columns. Substitutions of certain variables may be used as URLs for other entries in the table. Different data formats are recognized and processed according to their suffixes, *i.e.* a ".gif" or ".jpg" suffix is interpreted as a picture and rendered as such (*cf.* Figure 1 for an example).
  - Results may also be fed into selection lists, radio boxes, or check lists. Thus, query results can provide the initial setting of further HTML form fields.
- As a personalization feature of our web portal users store queries by assigning a personal label. All stored queries can be accessed through a selection list, to restart the query and retrieve the most up to date answers. This list of stored queries provides individual short cuts to often needed information.

5 Related Work

Our work combines approaches from different areas and extends these concepts in many directions. We here just give a short survey of this related work — a more detailed comparison may be found in (Staab *et al.* 2000).

Portals: Typically, portals like Yahoo!^{TM} are indices of web pages that are maintained by editors that manually classify web documents into a tree-like taxonomy of topics. In contrast to our approach those portals only utilize a very lightweight ontology that solely consists of categories arranged in a hierarchical manner. Due to its weak ontology Yahoo! cannot extend given information with facts derived by ontological axioms.

A community focused and ontology-based portal is RiboWeb (Altmann *et al.* 1999) that offers ribosome data and computational models for their processing. RiboWeb specifies ontologies in OKBC (Chaudri *et al.* 1998). The primary source of data is given by scientific literature which is manually linked to the different ontologies. Both systems, RiboWeb and our community portal, rely on ontologies for offering a semantic-based access to the stored data. However, the OKBC component of RiboWeb does not support the kind of automatic deduction that is offered by the inference engine of Ontobroker. Furthermore, RiboWeb does

not include wrappers for automatically extracting information from the given published articles. On the other hand, the computational modules of RiboWeb offer processing functionalities that are not part of (but also not intended for) our community web portal.

Database approaches: STRUDEL (Fernandez *et al.* 1998) and Hyperwave (Maurer 1996) apply concepts from database management systems to the process of building Web sites. They use database queries to generate web pages from database contents — allowing for multiple views onto the same content. When compared to our approach, these systems lack the semantic level that is provided in our approach by the domain ontology and the associated inference engine.

Knowledge representation for the web: The Ontobroker project (Decker *et al.* 1999) lays the technological foundations for the KA2 portal. Similar to Ontobroker are SHOE (Luke *et al.* 1997) and WebKB (Martin & Eklund 1999). All three systems aim at providing intelligent access to Web documents (though, with different means). However, they all lack an environment of methods and tools that are needed to build a community portal on their top and, thus, to make an application out of a core technology.

From our point of view, our community portal system is a rather unique AI application with respect to the collection of methods used and the functionality provided. Our approach for accessing information, providing information and maintaining the portal are more comprehensive than those found in other portals. We are able to offer this functionality since our backbone system Ontobroker and its add-ons provide more powerful techniques for *e.g.* inferencing or extracting information from various sources than those offered by comparable systems.

6 Conclusion

We have demonstrated in this paper how a community may build a community web portal. The portal is centered around an ontology that structures information for the purpose of presenting and provisoning information, as well as for the development and maintenance of the portal. We have described a particular application, the KA2 community web portal, that illustrates some of our techniques and methods.In particular, we have developed a set of ontology-based tools that allow to present multiple views onto the same information appropriate for browsing, querying, and personalizing web pages. Queries are responded to by an inference engine for F(rame)-Logic that integrates knowledge from many different sources.

For the future we are planning to integrate several semi-automatic information extraction-based approaches supporting the information provisoning part of our portal framework. Ontology revision and maintenance and the impact on the facts stored in the knowledge warehouse are currently not well understood and have to be researched further.

References

Altmann, R.; Bada, M.; Chai, X.; Carillo, M. W.; Chen, R.; and Abernethy, N. 1999. RiboWeb: An Ontology-based System for Collaborative Molecular Biology. *IEEE Intelligent Systems* 14(5):68–76.

Benjamins, R.; Fensel, D.; and Decker, S. 1999. KA2: Building Ontologies for the Internet: A Midterm Report. *International Journal of Human Computer Studies* 51(3):687.

Chaudri, V.; Farquhar, A.; Fikes, R.; Karp, P.; and Rice, J. 1998. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings 15th National Conference on Artificial Intelligence (AAAI-98)*, 600–607.

Decker, S.; Brickley, D.; Saarela, J.; and Angele, J. 1998. A Query and Inference Service for RDF. In *Proceedings of the W3C Query Language Workshop (QL-98), December 3-4*.

Decker, S.; Erdmann, M.; Fensel, D.; and Studer, R. 1999. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In Meersman, R., et al., eds., *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic Publisher. 351–369.

Erdmann, M., and Studer, R. 1999. Ontologies as Conceptual Models for XML Documents. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modelling and Mangement (KAW'99), Banff, Canada, October*.

Fernandez, M.; Florescu, D.; Kang, J.; and Levy, A. 1998. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of the 1998 ACM Int. Conf. on Management of Data (SIGMOD'98) , Seattle, WA*.

Kifer, M.; Lausen, G.; and Wu, J. 1995. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM* 42.

Lamping, L.; Rao, R.; and Pirolli, P. 1995. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*.

Luke, S.; Spector, L.; Rager, D.; and Hendler, J. 1997. Ontology-based Web Agents. In *Proceedings of First International Conference on Autonomous Agents*.

Maedche, A.; Schnurr, H.-P.; Staab, S.; and Studer, R. 2000. Representation language-neutral modeling of ontologies. In *Modellierung-2000: Proceedings of the German Workshop on Modeling. Koblenz, April, 2000*. Fölbach-Verlag.

Martin, P., and Eklund, P. 1999. Embedding Knowledge in Web Documents. In *Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999*. Elsevier Science B.V.

Maurer, H. 1996. *Hyperwave. The Next Generation Web Solution*. Addison Wesley.

Sahuguet, A., and Azavant, F. 1999. Wysiwyg Web Wrapper Factory (W4F). Technical report. http://db.cis.upenn.edu/DL/WWW8/index.html.

Staab, S.; Angele, J.; Decker, S.; Erdmann, M.; Hotho, A.; Maedche, A.; Studer, R.; and Sure, Y. 2000. Semantic Community Web Portals. In *Proceedings of the 9th World Wide Web Conference (WWW-9), Amsterdam, Netherlands*.

W3C. 1999. RDF Schema Specification. http://www.w3.org/TR/PR-rdf-schema/.

Wiederhold, G., and Genesereth, M. 1997. The Conceptual Basis for Mediation Services. *IEEE Expert / Intelligent Systems* 12(5):38–47.